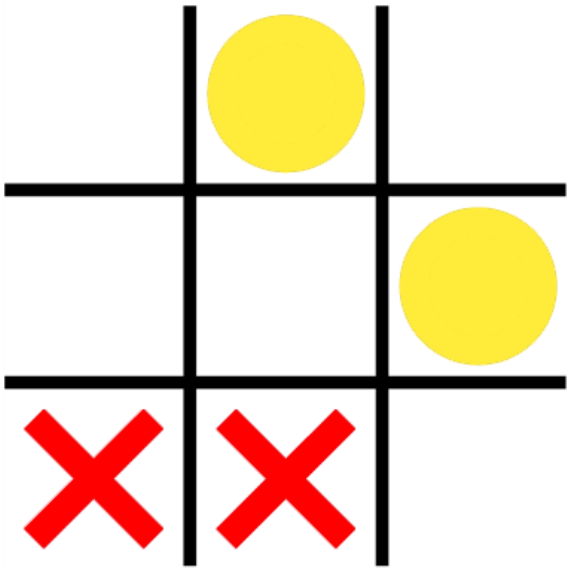
# Morpion



Le jeu Tic Tac Toe (jeu du morpion) se pratique à deux joueurs en tour par tour. Le but du jeu est d'être le premier joueur à créer un alignement sur la grille. Chaque joueur dispose de son jeton : croix rouge pour l'humain et rond jaune/orangé pour l'IA. Le support du jeu est une grille 3x3. Les alignements possibles de trois formes peuvent s'effectuer en vertical, en horizontal ou sur les diagonales.

Ouvrez le projet et lancez la partie. L'interface du jeu réagit au clic de la souris sur la grille. Vous pouvez ainsi disposer plusieurs marqueurs.

Conventions :

- Le joueur 1 est un joueur humain (c'est vous !!!)
- Le joueur 2 est une IA
- Un tableau 2D dénommé « Cases » stocke le contenu de chaque case
- Le code 0 signifie que la case est vide
- Lorsque le joueur 1 choisit la case (x,y), la valeur Cases[x][y] passe à 1
- Lorsque le joueur 2 choisit la case (x,y), la valeur Cases[x][y] passe à 2
- La partie est nulle et terminée si toutes les cases sont remplies et qu'il n'y a aucune configuration gagnante.
- Si un joueur remporte la partie, son score augmente de 1

**Consignes : choisissez des noms de variables clairs et créez une fonction par tâche / traitement / opération…**

# Etape 1 - Gagné

Cette première étape consiste à mettre en place le gestionnaire de parties : placement des jetons du joueur humain, détection d'une partie gagnante, fin de partie avec affichage du gagnant et changement du score, puis relance d'une nouvelle partie.

**Configuration gagnante** : après le placement d'un jeton par le joueur humain, détectez si une configuration gagnante est apparue. Dans ce cas, appliquez la procédure de fin de partie.

**Affichage fin de partie** : modifiez le score et changez la couleur de la grille pour qu'elle corresponde à la couleur du joueur gagnant : rouge ou jaune.

A ce niveau, il n'y a qu'un joueur dans la partie, ce n'est donc pas très dur de gagner ☺

# Etape 2 – Perdu / Match nul

**Joueur IA** : pour l'instant, l'IA choisit une position au hasard parmi les cases disponibles.

**Configurations Perdu / Match nul** : détectez maintenant un placement gagnant de l'IA. Détectez aussi la présence d'un match nul : aucun joueur gagnant et aucune case disponible. Dans ce cas, l'affichage en fin de partie présentera une grille blanche..

**Remarque** : Tout d'abord, le joueur humain commence la partie. Le clic souris sur une case vide déclenche le positionnement d'un pion du joueur. On effectue les tests nécessaires pour savoir si la partie est terminée. Dans le cas contraire, c'est à l'IA de jouer. On effectue une nouvelle fois les tests pour savoir si la partie est terminée. A ce niveau, le programme n'effectue plus d'actions. En effet, c'est au joueur humain de jouer (nouvelle partie ou partie en cours). Son tour de jeu sera déclenché par un clic de la souris. C'est donc la fonction ClicSouris appelée par le système qui va déclencher le tour suivant. Afin de nous y retrouver, nous utilisons une variable d'état DébutDePartie indiquant qu'une nouvelle partie va commencer :

Fonction ClicSouris(…)

- Si DébutDePartie  // lancement d'une nouvelle partie
    - Remise à zéro des cases
    - Affichage de la grille (bleu par défaut)
    - DébutDePartie  = Faux
- Si la case cliquée contient déjà un pion => quittez
- Placement du pion du joueur humain
- Si partie  terminée :
    - DébutDePartie  = Vrai
    - Affichage fin de partie
- Sinon
    - Placement du pion du joueur IA
    - Si partie terminée
        - DébutDePartie  = Vrai
        - Affichage fin de partie

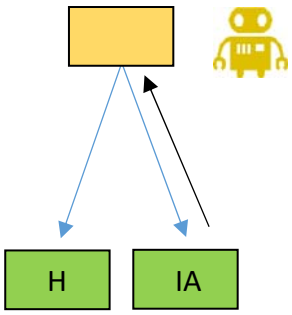
# Etape 3 – L'IA

On se propose de mettre en place un algorithme de simulation qui permet, dans le cas du morpion, de fournir un joueur IA excellent. S'il peut gagner, il le fera !

L'algorithme se lance à chaque fois que l'IA doit jouer. Il analyse la meilleure stratégie à développer. Il est sans mémoire, c'est-à-dire, qu'il ne tient pas compte du style de jeu du joueur humain et des coups précédemment joués. Lorsqu'il est lancé, il a pour seules données : l'état de la grille de jeu et les coups pouvant être effectués. S'il sait où jouer, cela sous-entend qu'il « connait » l'ensemble des règles du jeu.

Prenons un cas pratique. Le joueur Humain vient de jouer. L'algorithme est donc lancé pour déterminer quel coup l'IA va jouer. Mais comment savoir quel coup est le meilleur ? Pas facile. Cependant, l'algorithme sait quels sont les coups possibles. Alors, il va « simuler » toutes les parties possibles en partant de la grille actuelle.

Par exemple, supposons qu'en choisissant l'option de gauche, l'algorithme détermine que l'IA va perdre ce qui veut dire que l'humain (H) va gagner. L'algorithme examine alors l'option de droite et les simulations indiquent que cette option permettrait à l'IA de gagner. Quel choix faire ? Comme c'est au tour de l'IA de jouer et qu'elle peut librement choisir entre les deux options, elle va donc retenir la plus avantageuse. Ainsi, l'IA choisit l'option de droite lui permettant de gagner.



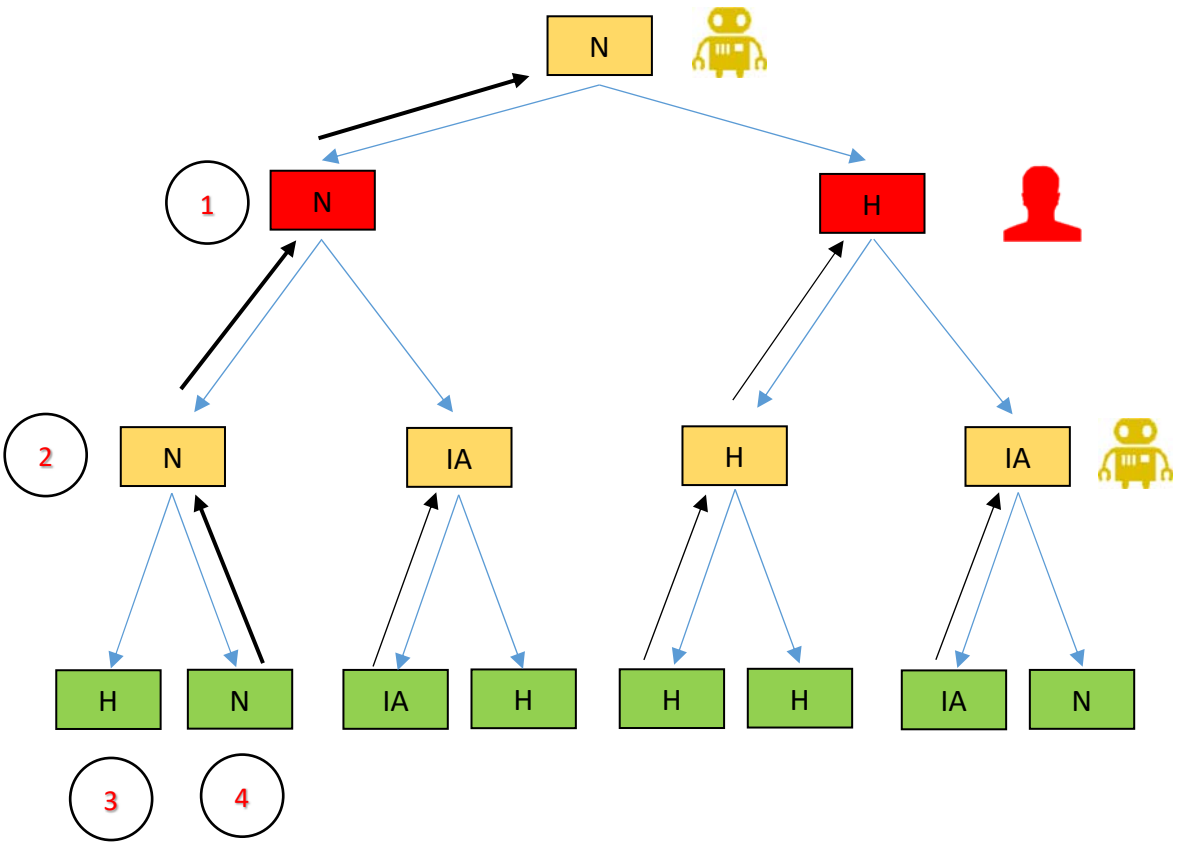

Il existe une difficulté, en effet, il y a trois issues possibles à une partie : Perdu, Gagné et Nul. On suppose que tous les joueurs préfèrent un match nul à un match perdu et un match gagné à un match nul. Lorsqu'une partie se termine durant la simulation, on va remonter le nom du joueur gagnant : H pour humain, IA pour l'IA et N pour un match nul.

Comment savoir si une partie se termine ? Pour une grille donnée, on peut déterminer si l'humain a gagné ou si l'IA a gagné, dans ce cas là la partie est terminée. Si aucun des deux joueurs n'a gagné et qu'il reste des cases libres, la partie continue. Si toutes les cases sont remplies et qu'aucun des deux joueurs n'a gagné, lar partie est terminée et nous sommes en présence d'un match nul.

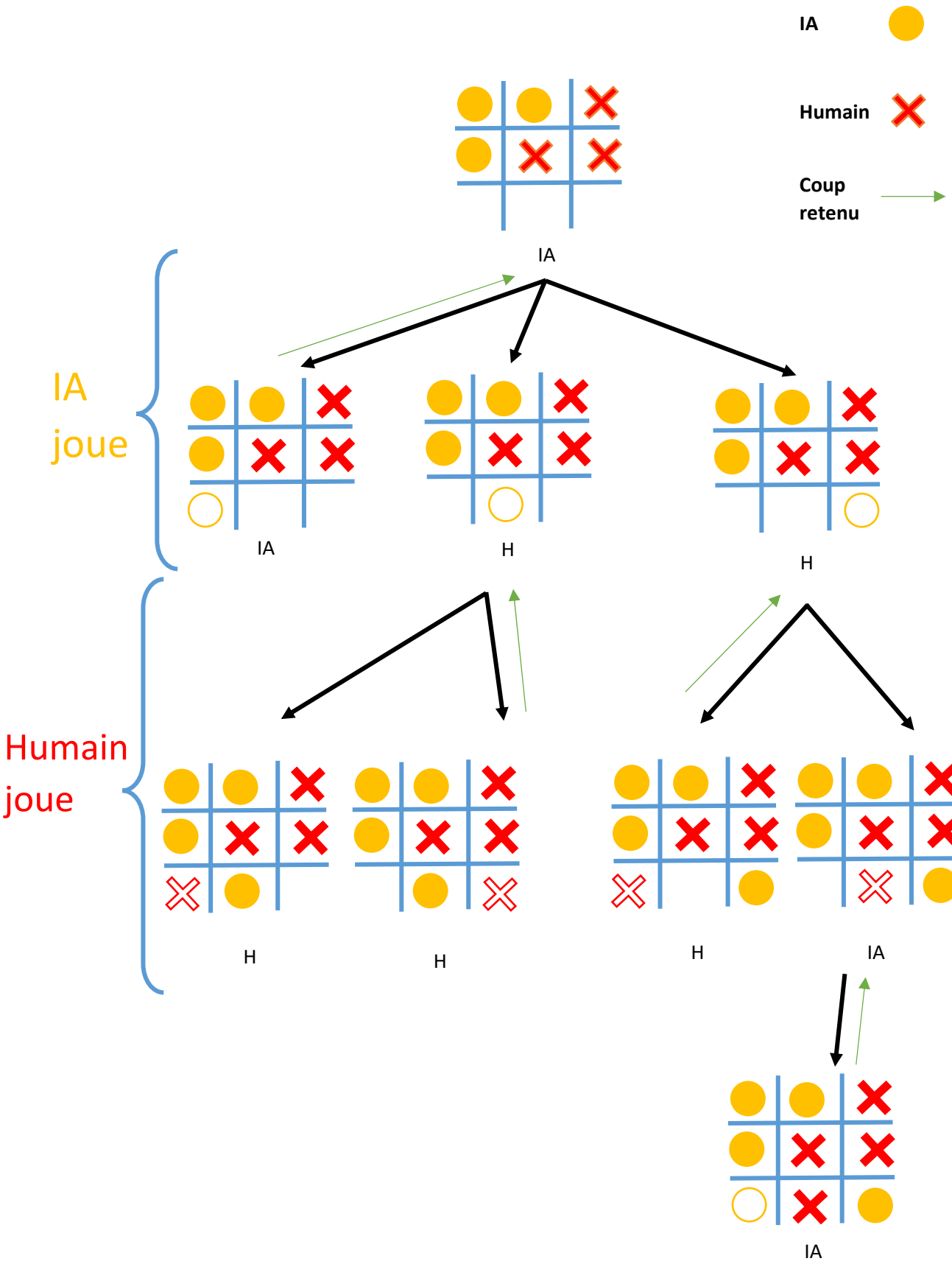
Nous avons expliqué comment l'IA fait son choix, mais comment l'algorithme a-t-il fait cette mystérieuse simulation qui amène à la connaissance des meilleurs scores de partie ? En fait, l'algorithme simule le joueur humain ainsi que le joueur IA.

Ainsi durant la simulation, si c'est à l'IA de jouer, l'algorithme va chercher à maximiser le gain de l'IA et si c'est à l'humain de jouer, l'algorithme va chercher à maximiser le gain de l'humain. La simulation est un algorithme récursif, c'est-à-dire qu'il s'appele lui-même pour résoudre le problème. La récursion s'arrête lorsque la partie se termine. A ce moment, l'issue de la partie est connue et retournée à la fonction appelante.

Prenons un exemple où les simulations s'achèvent au bout de trois coups, cela simplifie le schéma ☺ L'algorithme démarre son cheminement en haut du schéma. Il détermine alors que deux coups sont possibles. Il commence par simuler le premier. Nous sommes ici au niveau du marqueur 1. L'algorithme va ensuite simuler les parties possibles après avoir joué ce coup. Pour cela, l'algorihme détermine les coups possibles et en trouve 2. Il choisit le premier coup pour continuer son exploration et nous passons au marqueur 2. L'algorithme simule alors les deux coups possibles (marqueurs 3 et 4). Comme ces coups terminent la partie en cours, ils sont associés au résultat H pour le premier et N pour le second. Au niveau de la position 2, c'était au tour du joueur IA de jouer. L'algorithme va donc retenir la meilleure issue que l'AI peut espérer : un match nul N. Au niveau du marque 1, les deux options possibles sont N et IA, comme l'algorithme joue pour le joueur humain à ce niveau, il va retenir l'option la plus favorable à l'humain c'est-à-dire : N. Ainsi de suite…



Nous allons présenter le déroulement de la récursivité pour le jeu du morpion :



Voici la logique :

```
JoueurSimuleIA() :
    Si la partie est finie
        retourner Résultat
    L = liste des coups possibles (les cases vides)
    Résultats = [ ]
    Pour chaque coup K dans L
        Jouer le coup K  (on pose le pion)
        R = JoueurSimuleHumain()
        Stocker R et K dans Résultats
        Annuler le coup K (on retire le pion)

    Retourner le coup et le résultat le plus intéressant pour l'IA


JoueurSimuleHumain() :
    Si la partie est finie
        retourner Résultat
    L = liste des coups possibles (les cases vides)
    Résultats = [ ]
    Pour chaque coup K dans L
        Jouer le coup K  (on pose le pion)
        R = JoueurSimuleIA()
        Stocker R et K dans Résultats
        Annuler le coup K (on retire le pion)

    Retourner le coup et le résultat le plus intéressant pour l'Humain
```

Partiefinie() :

- Joueur IA a gagné
- Joueur Humain a gagné
- Plus de cases disponibles pour jouer

# Projet : Puissance 4

Le but du jeu est d'aligner une suite de 4 pions de même couleur sur une grille comptant 6 rangées et 7 colonnes. Chaque joueur dispose de 21 pions d'une couleur (par convention, en général jaune ou rouge). Tour à tour les deux joueurs placent un pion dans la colonne de leur choix, le pion coulisse alors jusqu'à la position la plus basse possible dans ladite colonne à la suite de quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le premier un alignement (horizontal, vertical ou diagonal) consécutif d'au moins quatre pions de sa couleur. Si, alors que toutes les cases de la grille de jeu sont remplies, aucun des deux joueurs n'a réalisé un tel alignement, la partie est déclarée nulle.

Nous vous conseillons à partir de cette étape de dupliquer le source du jeu du morpion car nous allons nous en servir pour créer le jeu du puissance 4.

## Etape 1 - Jeu minimal

Modifier le jeu du morpion pour gérer une grille de 6x7.

Modifier l'affichage pour que le joueur 1 dispose des jetons jaunes et le joueur 2 des jetons rouges.

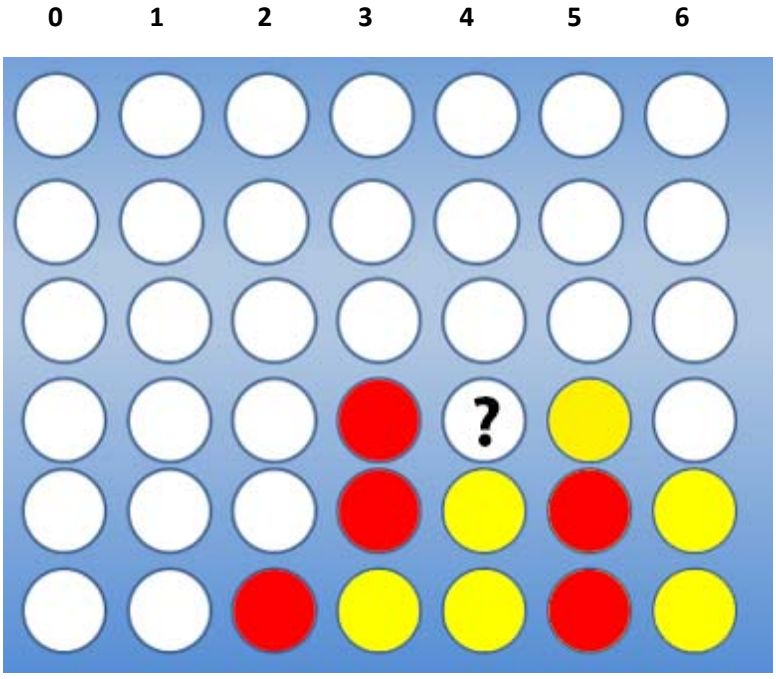
Mettre en place une fonction qui détecte si un alignement de 4 pions du même joueur existe. Le numéro du joueur sera un paramètre de la fonction. Les alignements peuvent comme dans le cas du morpion être horizontaux, verticaux ou diagonaux. Cette fonction n'est pas forcément simple à mettre en œuvre.

Mettre à jour la fonction qui détermine les coups possibles. Nous vous proposons de coder un coup possible par son numéro de colonne. Ainsi en début de partie la liste des coups possibles sera égale à [0, 1, 2, 3, 4, 5, 6]. Dès qu'une colonne est pleine, son numéro va disparaître de la liste des coups possibles.

Nous sommes donc à cette étape capable de mettre en place un jeu contre une IA. Nous pouvons tout simplement reprendre l'IA qui consiste à choisir un coup au hasard parmi les coups possibles. Cela vous permet de tester le bon fonctionnement du jeu et surtout de valider la détection des coups gagnants.

# Etape 2 - Placements judicieux

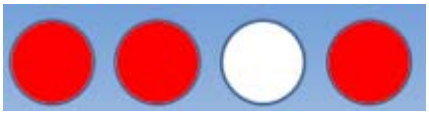
Nous allons essayer de fournir une IA plus perspicace qu'un simple choix aléatoire. Pour cela, nous vous proposons d'associer un score suivant les opportunités fournies par un placement. Par exemple, lorsque nous avons la grille suivante et que le joueur rouge doit jouer, nous aurions plus tendance à placer notre pion sur la colonne 4 car il permet de construire une ligne diagonale de trois pions rouges. Mais finalement, jouer sur la colonne 5 semble tout aussi intéressant car on place trois des quatre pions rouges nécessaires à un alignement. Joueur sur la colonne 6 ne semble pas fournir d'intérêt pour former un alignement rouge, par contre, il permet de bloquer un début d'alignement jaune composées de 2 pions pour l'instant.



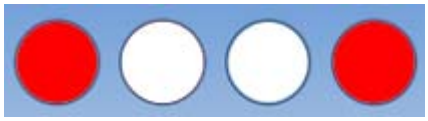

Jouer sur la colonne 6 s'avère cependant un mauvais choix, car cela permet au joueur jaune de créer un alignement en jouant sur la colonne 6. Nous ne prendrons pas en compte cela pour l'instant. Nous allons juste considérer les avantages/désavantages produits immédiatement par le placement d'un pion. Nous vous proposons les scores suivants, vous pouvez les faire évoluer à votre guise.

- Si le placement permet de faire un alignement de 4 => 100 points
- Si le placement bloque un alignement de 3 pions du joueur adverse => 50 points
- Si le placement permet de faire un alignement de 3 sur 4 => 30 points
- Si le placement bloque un alignement de 2 sur 4 pions du joueur adverse => 15 points
- Si le placement permet de faire un alignement de 2 sur 4 => 10 points

Attention, on considère uniquement le score max, on ne cumule pas les différents scores. Car on ne veut pas qu'un placement qui permette de bloquer 2 alignements de 3 pions du joueur adversaire passe devant le score d'un alignement possible de 4 pions pour le joueur courant.

 Le placement disponible obtient un score de 100 points

Mais dans la même logique !

 Les deux placements disponibles obtiennent un score de 30

Car ce qui est important c'est le nombre de pions présent sur les 4 emplacements. Le fait qu'ils ne soient pas contigus n'est pas une raison pour ne pas comptabiliser de score.

Travail à effectuer : réutiliser la fonction qui détermine les colonnes où il est possible de joueur. Pour chaque colonne disponible, trouver l'emplacement correspondant et calculez ses points associés. Pour l'ensemble des colonnes, trouver celle associée au placement de meilleur score. Faîtes évoluer votre IA pour qu'elle place son pion sur la colonne retenue.

A ce niveau, si vous voulez vous amuser, vous pouvez lancer un challenge entre l'IA qui joue au hasard et l'IA qui cherche un emplacement judicieux. Sur 10 parties, quel est le ratio de réussite ?

# Etape 3 - Minimax

Nous avons vu tout à l'heure pour le jeu du morpion que l'algorithme du minimax permet de trouver la meilleure solution. En effet, il simule TOUTES les parties pouvant être jouées à partir d'une grille de jeu et suivant une séquence de max alternés détermine la meilleure option. Ainsi, au début de la partie de morpion, la simulation a 9 choix pour poser son premier jeton. Une fois posée, elle en a 8 pour le joueur suivant, puis 7 encore pour le joueur d'après. Cela nous amène à un total de 9! simulations soit 9x8x7x6x5x4x3x2 = 362 880 parties à analyser. Sur un ordinateur récent avec une vitesse de l'ordre du GigaHertz, cette exploration est possible dans un temps acceptable.

Cependant, dans le cas du puissance 4, le ton change. En effet, il y a 42 tours de jeu possibles et un nombre de parties différentes évaluées à environ 4 531 milliards. Nous avons donc un jeu 10 millions de fois plus complexes que le morpion ! L'algorithme développé pour le morpion ne pourrait pas trouver la solution dans un temps acceptable.

Nous allons cependant appliquer l'approche du minimax mais en restreignant la profondeur d'exploration. Pour cela, nous allons simuler toutes les parties à partir de la grille actuelle en jouant au maximum 2 coups. Certes, la partie ne sera pas forcément finie et nous n'aurons donc pas le score associé à perdu/gagné/nul. Mais si l'on examine l'algorithme minimax, tout ce qu'il requiert pour prendre une décision est finalement un score ou une sorte de « note ». On peut donc comme d'habitude si la partie se termine retourner le score associé mais si la partie n'est pas finie on peut se contenter de retourner une note qui caractérise la grille de jeu. Plus cette note sera haute, plus elle sera à l'avantage du joueur IA et plus elle sera basse, plus elle sera à l'avantage du joueur humain. L'algorithme Minimax va donc prendre une décision de jeu qu'il sait nous amener vers une configuration de jeu de « bonne » qualité.

Nous devons donc créer une fonction retournant une évaluation de la grille actuelle. Si la partie n'est pas gagnante/perdante/nulle, c'est qu'il n'y a aucun alignement de 4 pions. Nous vous proposons de de réutiliser la fonction qui note les placements pour construire cette nouvelle fonction. Ainsi nous aurons :

```
Note(Grille)

    Si IA est gagnante retourner 500
    SI Humain est gagnant retourner -500
    LIA = []   # Liste des notes des emplacements pour le joueur IA
    LH  = []   # Liste des notes des emplacements pour le joueur Humain
    Pour chaque colonne où l'on peut joueur
        Calculer la position de l'emplacement libre
        Calculer le score de l'emplacement pour le joueur IA => LIA
        Calculer le score de l'emplacement pour le joueur Humain => LH
    Retourner Max(LIA) – MAX(LH)
```

Déterminez le maximum de coups que l'on peut simuler pour avoir une réponse de l'IA qui prend moins d'une seconde ?

Mettez en place un challenge entre l'IA qui examine uniquement les placements judicieux (question précédente) et l'IA avec un minimax à profondeur limitée. Y-a-t-il un réel gain sur la stratégie ?

# Etape 4 – 2ème IA

Ajoutez un troisième joueur dans la partie. Cet IA aura des pions verts et son objectif sera de construire un carré 2x2 pour gagner. Toute autre idée originale est la bienvenue.